Machine Learning Research
2024; 9(2): 64-74
http://www.sciencepublishinggroup.com/j/mlr
doi: 10.11648/j.mlr.20240902.15
ISSN: 2637-5672 (Print); ISSN: 2637-5680 (Online)

Modified XGBoost Hyper-Parameter Tuning Using Adaptive Particle Swarm Optimization for Credit Score Classification

Kenneth Kiprotich Langat[1, 2, *], Anthony Gichuhi Waititu[3], Philip Odhiambo Ngare[4]

[1]Department of Mathematics, Pan African Institute of Basic Science Technology and Innovation, Nairobi, Kenya
[2]Department of Mathematics, Egerton University, Nakuru, Kenya
[3]Department of Statistics and Actuarial Sciences, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya
[4]Department of Mathematics, University of Nairobi, Nairobi, Kenya

Email address:
langatken26@gmail.com (Kenneth Kiprotich Langat)
[*]Corresponding author

To cite this article:
Kenneth Kiprotich Langat, Anthony Gichuhi Waititu, Philip Odhiambo Ngare. (2024). Modified XGBoost Hyper-Parameter Tuning Using Adaptive Particle Swarm Optimization for Credit Score Classification. *Machine Learning Research, 9*(2), 64-74. https://doi.org/10.11648/j.mlr.20240902.15

Received: 24 August 2024; **Accepted:** 21 September 2024; **Published:** 31 October 2024

Abstract: In recent years credit scoring has become a challenging issue among financial institutions. Several researchers have dedicated efforts in machine learning in the areas of credit scoring and results have shown that machine learning algorithms have had a satisfactory performance in the sector of credit scoring. Decision trees have been used for data sets that have high dimension and have a complex correlation and the benefits of feature combination and feature selection has led to the usage of decision trees in classification. The disadvantage of decision tree which is overfitting has led to the introduction of extreme gradient boosting that overcomes the shortcoming by integrating tree models. Employing optimization method helps in tuning the hyperparameters of the model. In this paper, a modified XGBoost model is developed that incorporates inflation parameter. In addition to the proposed model, the study uses adaptive particle swarm optimization since it does not fall into local optima. The swarm split algorithm uses clustering and two learning strategies to promote subswarm diversity and avoid local optimums. In this study the modified XGBoost model was compared to five traditional machine learning algorithms namely, the standard XGBoost model, logistic regression, KNN, support vector machine and decision tree. The study used one data set in credit scoring and the evaluation measures used were accuracy, precision, recall and F1-score. Results demonstrate that the proposed model outperforms other models.

Keywords: Modified XGBoost, Credit Scoring, Optimization

1. Introduction

Every financial institution knows the importance of knowing the creditworthiness of their borrowers and thus the need of great understanding of credit scoring process. It requires the lender to assess the risk(s) related to advancing loan to a client. The management of financial institution majorly rely on the positive performance of the scorecard model. For decades, several techniques of improving the performance of credit scoring models has been developed, among these are the recent machine learning and artificial intelligence algorithms with their various optimization methods.

Application scoring is the process of determining the credit quality of new applicants. It assesses the risk of credit requests based on their age, income, and occupation. This topic is essentially a categorization task for the general population [13]. In other words, a classification model (also known as a classifier) is required to classify credit requests as good or bad credit depending on their features. As a result, different categorization techniques have been used, which are grouped into two major categories: statistical techniques and artificial intelligence (AI) techniques [4].

The traditional models for credit scoring heavily depend on statistical techniques. However, in the recent past years, technology have brought a lot of improvement such that there are more advanced models like neural network, random forest, ensemble learning models et cetera that are now taking over. [8] through their study showed that support vector machines (SVM) is more accurate than the traditional methods when creating a credit scoring model. Another research by Hens and [6] combined SVM and stratified sampling technique to bring down the computational time but retaining its accuracy at a higher level.

The combination of two or more models (ensemble learning) as a way of boosting performance accuracy of the model has attracted many researchers in the recent past. [12] carried out research to compare ensemble learning methods for credit scoring and found that ensemble learning models continuously outshine the single models. On the other side, [7] proved the robustness and adaptability of ensemble learning method to different datasets by constructing a data-driven ensemble classifier model for credit scoring.

The latest advancements in credit scoring have shift to bring on board economic factors either has parameter(s) or as variable and at the same time using optimization techniques to upscale model performance. This can be seen in the work of [15] who developed a cost-sensitive boosted tree for loan evaluation, in order to manage the issue of imbalanced data in credit scoring. Moreover, [3] came up with XGBoost model, which is a scalable tree boosting system.

Incorporating economic indicators such as inflation rates, gross domestic product etc into credit scoring models is one way of making the model more comprehensive when it comes to evaluation of credit risk. The application of these economic indicators is mostly good enough to be applied during the economic shock periods like when the inflation has gone up because of things like war or politics which directly affect the ability of loan applicant to repay loan. The proposed model in this study modifies the XGBoost algorithm by including inflation rates as a parameter.

Hyperparameter tuning is a very important step in optimizing the performance of machine learning models. Traditional methods like greedy search and random search have been widely used for this purpose. [1] gave the merits of random search over greedy search in high-dimensional spaces. The problem with these methods is that it is expensive in terms of computation and at the same time might not yield the best results.

Particle Swarm Optimization (PSO) is one of the optimization techniques inspired by the social behavior of birds flocking or fish schooling. [10] confirmed PSO to be an efficient optimizer for different applications. The adaptive version of PSO (APSO) further improved the algorithm's performance by dynamically varying the learning parameters based on the swarm's behavior. This study utilizes APSO to find the optimal hyperparameters for the modified XGBoost model, ensuring improved credit scoring accuracy.

Many recent studies have explored the use of optimization techniques in machine learning models for credit scoring. [5] suggested a two-stage fuzzy neural approach for credit risk assessment and illustrate the possibility of hybrid models in upgrading credit scoring performance. Similarly, [2] applied eXtreme Gradient Boosting (XGBoost) to construct credit risk assessment models for financial institutions, showcasing its effectiveness in handling big data.

A study by [9] compared traditional machine learning algorithms, popular ensemble learning classifiers, and four hyperparameter optimization methods: grid search, random search, tree-structured Parzen estimator, and particle swarm optimization. Experiments were conducted on four credit datasets and seven KEEL benchmark datasets using five popular evaluation metrics: accuracy, error rate (type I and II), Brier score, and F1 score. The results show that the proposed model outperforms the other models on average.

A study by [17] used XGBoost for bankruptcy prediction. The XGBoost model effectively evaluates uneven data from Polish enterprises by using an AUC measure and forcing optimal data sorting.The model outperforms other methods like NN, LR, SVM, and RF.

The authors in [11] created an ensemble model that combines the AdaBoost approach with the NN base classifier and uses PSO to find the best connection weight for the NN.The results indicate that this model outperforms other models in processing German and Australian datasets.The original PSO algorithm was developed to optimize a continuous space, as the particle state and motion rules are continuous real numbers [16]. .PSO, rather than GS, RS, and TPE, is more suited for hyperparameter optimization in XGBoost. PSO converges quickly and iteratively finds the optimal solution.

Furthermore, adaptive particle swarm optimization outperforms traditional hyperparameter optimization algorithms.

The No Free Lunch Theorems for optimization, introduced by [14], emphasize that no single optimization algorithm performs best across all problems. This underline the importance of choosing an appropriate optimization method depending on the specific characteristics of the problem.

The integration of inflation rates into credit scoring models and the use of advanced optimization techniques like APSO represent significant advancements in the field of credit risk assessment. These innovations not only enhance the predictive accuracy of credit scoring models but also ensure their robustness under varying economic conditions. Future research should continue to explore the potential of combining different machine learning algorithms and optimization techniques to further improve credit scoring methodologies.

2. Methodology

2.1. Modified XGBoost

XGBoost is one of the competent ensemble learning methods for both regression and classification. Modified XGBoost is an improved version of XGBoost where by the inflation parameter as been incorporated and its performance proof to be much better than for the normal XGBoost. It uses

Taylor expansion method to approximate the loss function. The following is the description of the modified XGBoost.

Based on the objective function of the normal Extreme Gradient Boosting model given by;

$$\min L^t(y,\hat{y}^t) = \min\Big(\sum_{i=1}^{n} l(y_i,\hat{y_i}^t) + \Omega(f_t)\Big) \tag{1}$$

Where,
The model entails;
n - Total number of samples (loan clients)
m - Number of variables/features
z_i - variable information of the i^{th} sample, $z_i \in \mathbb{R}^m$
y_i - The value of the i^{th} sample
$\hat{y_i}$ - The predicted value of the i^{th} sample
$\hat{y}_i^t$ - The predicted value up to the t^{th} tree
$l(y_i,\hat{y_i})$ - The loss function of the i^{th} sample
$L(y,\hat{y})$ - The loss function of total sample
$\Omega(f_k)$ - Regularization term of objective function to prevent overfitting, f_k represent the k^{th} decision tree
$D = \{(z_i, y_i | z_i \in \mathbb{R}^m, y_i \in \mathbb{R}, z_i = \{_{i,1}, z_{i,2}, ..., z_{i,m} | i = 1, 2, ..., n\})\}$

The study replaces the standard regularization function $\Omega(f_t)$ with the modified regularization function $\Omega^*(f_t)$ and so equation (1) becomes;

$$\min L^{*t}(y,\hat{y}^t) = \min\Big(\sum_{i=1}^{n} l(y_i,\hat{y_i}^t) + \Omega'(f_t)\Big) \tag{2}$$

Where;

$$\Omega^*(f_t) = \gamma T_t + \frac{1}{2}(\lambda + 2\pi)\sum_{j=1}^{T_t} w_{t,j}^2$$

and

$$f_t(z_i) = w_t q(z_i)$$

Where:
T_t - Number of leaf nodes in the t^{th} tree
γ - Contraction (pruning) coefficient of the number of leaf nodes
$w_{t,j}$ - the score of the j^{th} leaf node in the t^{th} tree
λ - penalty coefficient of the score of leaf nodes.
π - Inflation rate

The study use the modified Ridge regularization and so equation (2) becomes;

$$\begin{aligned}\min L^{*t}(y,\hat{y}^t) &= \min\Big(\sum_{i=1}^{n} l(y_i,\hat{y_i}^t) + \Omega^*(f_t)\Big)\\ &= \min\Big(\sum_{i=1}^{n} l(y_i,\hat{y_i}^t) + \frac{1}{2}(\lambda+2\pi)\sum_{j=1}^{T_t} w_{t,j}^2 + \gamma T_t\Big)\end{aligned} \tag{3}$$

2.2. Optimizing the Modified Model

Assuming that data is give as follows, $(z_1, y_1), (z_2, y_2), (z_3, y_3), ..., (z_n, y_n)$ where z_i represents the independent variable and y_i represents the dependent variable. The optimization steps are given as:

$$\hat{y_i}^t = \sum_{k=1}^{t} f_k(z_i) = \hat{y_i}^{(t-1)} + f_t(z_i) \tag{4}$$

where $\hat{y_i}^t$ is the predicted value of the model in the round t and the proposed Modified XGBoost model algorithm is formed by continuous iteration, and each iteration is trained by adding a lesson of decision tree to the prediction value $\hat{y_i}^t$ of the previous round.

Generally, the formula for the objective function is:

$$obj(\phi) = L(\phi) + \Omega(\phi) \tag{5}$$

where ϕ is the parameter to be estimated, $L(\phi)$ is the loss function and $\Omega(\phi)$ is the regularization term. Thus, we minimize $obj(\phi)$ which gives the criterion for selecting $f(x)$

$$\begin{aligned}L^{*t} &= \sum_{i=1}^{n} l\Big(y_i, \hat{y}_i^{(t)}\Big) + \Omega^*(f_t)\\ &= \sum_{i=1}^{n} l\Big(y_i, \hat{y}_i^{(t-1)} + f_t(z_i)\Big) + \Omega^*(f_t) + constant\end{aligned} \tag{6}$$

Taylor expansion is used to expand the approximate objective function and remove the constant term. Thus equation (6) become;

$$\begin{aligned} L^{*t} &= \sum_{i=1}^{n} \Big[g_i f_t(z_i) + \frac{1}{2} h_i f_t^2(z_i)\Big] + \Omega^*(f_t) \\ &= \sum_{i=1}^{n} \Big[g_i f_t(z_i) + \frac{1}{2} h_i f_t^2(z_i)\Big] + \gamma T_t + \frac{1}{2}(\lambda + 2\pi) \sum_{j=1}^{T_t} w_{t,j}^2 \end{aligned} \tag{7}$$

where:

$$g_i = \partial_{\hat{y}(t-1)} l(y_i, \hat{y}^{t-1})$$

$$h_i = \partial^2_{\hat{y}(t-1)} l(y_i, \hat{y}^{t-1})$$

Among the n samples, some will share same leaf node and thus making the summation to run from 1 to the number of leaf nodes in a tree.

The set of samples sharing same leaf is represent by, $I_j = \{i | q(z_i) = j\}$, which is in the j^{th} leaf node in the tree. $q(z_i)$ is a function that maps the samples z_i to the leaf j. Letting w to be the score of the leaf, then $f(x) = wq(z)$, $w \in \mathbb{R}^T, q : \mathbb{R}^d \longrightarrow \{1, 2, ..., T\}$

By letting,

$$f_t(z_i) = w_{t,j}$$

and

$$G_j = \sum_{i \in I_j} g_i$$

and

$$H_j = \sum_{i \in I_j} h_i$$

Then the final objective function is:

$$\begin{aligned} L^{*t} &= \sum_{j=1}^{T_t} G_j w_{t,j} + \Big[\frac{1}{2} \sum_{j=1}^{T_t} \Big(H_j + \lambda + 2\pi\Big) w_{t,j}^2\Big] + \gamma T_t \\ &= \sum_{j=1}^{T_t} \Big[G_j w_{t,j} + \frac{1}{2}\Big(H_j + \lambda + 2\pi\Big) w_{t,j}^2\Big] + \gamma T_t \end{aligned} \tag{8}$$

The optimal value of $w_{t,j}$ is obtained by differentiating equation (8) with respect to $w_{t,j}$ and equating to zero. That is;

$$\begin{aligned} 0 =& \frac{\partial}{\partial w_{t,j}} \Big[G_j w_{t,j} + \frac{1}{2}\Big(H_j + \lambda + 2\pi\Big) w_{t,j}^2 + \gamma T_t\Big] \\ =& \frac{\partial}{\partial w_{t,j}} G_j w_{t,j} + \frac{1}{2} \frac{\partial}{\partial w_{t,j}} \Big(H_j + \lambda + 2\pi\Big) w_{t,j}^2 + \frac{\partial}{\partial w_{t,j}} \gamma T_t \\ =& G_j + (H_j + \lambda + 2\pi) w_{t,j} \end{aligned} \tag{9}$$

Thus the optimal weight for each leaf of a tree is;

$$w_{t,j}^* = -\frac{G_j}{H_j + \lambda + 2\pi} \tag{10}$$

The final optimized objective function is attained by replacing $w_{t,j}$ in equation (8) by $-\dfrac{G_j}{H_j + \lambda + 2\pi}$, that is;

$$\begin{aligned} minL^{*t} &= \sum_{j=1}^{T_t}\Big[G_j(-\frac{G_j}{H_j+\lambda+2\pi}) + \frac{1}{2}(H_j+\lambda+2\pi)(-\frac{G_j}{H_j+\lambda+2\pi})^2\Big] + \gamma T_t \\ &= \sum_{j=1}^{T_t}\Big[-\frac{G_j^2}{H_j+\lambda+2\pi} + \frac{G_j^2}{2(H_j+\lambda+2\pi)})\Big] + \gamma T_t \\ &= -\frac{1}{2}\sum_{j=1}^{T_t}\frac{G_j^2}{H_j+\lambda+2\pi} + \gamma T_t \end{aligned} \tag{11}$$

The final objective function becomes:

$$\min L^{*t} = -\frac{1}{2}\sum_{j=1}^{T_t}\frac{G_j^2}{H_j+\lambda+2\pi} + \gamma T_t \tag{12}$$

2.3. Hyper-parameter Tuning

Results of any model in machine learning depends majorly on the parameters of the model. High quality parameters leads to good performance of the model. Parameters of the model depends on the dataset distribution while hyperparameter relies on the complexity of the model. Powerful set of hyperparameters contribute to improved performance of the training model and hence the need of tuning these hyperparameters. Among the old and known hyperparameter tuning algorithms are greedy search, random sampling and Bayesian.

Greedy search is the best but it only work with the limited space of hyperparameters and thus not good for models with large dimensions. Random sampling algorithm needs a specified distribution to work on in a specific space range. Table 1 below shows the hyperparameters to be tuned.

Table 1. *Hyperparameters of Modified XGBoost.*

Hyperparameter	Purpose	Range	Default value
Maximum depth	Control model complexity	$[0,\infty]$	6
Minimum child weight	Decision for further partitioning	$[0,\infty]$	1
Learning rate	Prevent overfitting	[0,1]	0.3
Gamma	For partitioning leaf node	$[0,\infty]$	0
Subsample ratio	Number of sample used at a given training	(0,1]	1
Column subsample ratio	Number of features used at each training	(0,1]	1

2.4. Particle Swarm Optimization (PSO)

It is an optimization algorithm that depends on the speed of the weightless particles and position which tells the direction that the particle is moving towards. The set of these particles is the search space for personal best position and global best position. Global best relies on the personal bests.

The algorithm is as below;

Let the search space be a K-dimensional with n particles, that is $(z_1, z_2, ..., z_n)$ and so $z_j = (z_{j1}, z_{j2}, ..., z_{jK})$. At time t, information about the particle z_j is as follows; position$Z_j^t = [z_{j1}^t, z_{j2}^t, ..., z_{jK}^t]^T$, Speed $S_j^t = [s_{j1}^t, s_{j2}^t, ..., s_{jK}^t]^T$, individual best location $l_i^t = [l_{i1}^t, l_{i2}^t, ..., l_{iK}^t]^T$ and global best location $l_g^t = [l_{g1}^t, l_{g2}^t, ..., l_{gK}^t]^T$ The two properties of a particle, speed and location keep on changing and so the two at time $t+1$ will be as below;

$$s_{ik}^{t+1} = \omega s_{ik}^t + c_1 r_1^t\left(l_{ik}^t - z_{ik}^t\right) + c_2 r_2^t\left(l_{gk}^t - z_{ik}^t\right) \tag{13}$$

Equation (11) is the formula for speed while the next is the position formula. Position formula depend on the speed formula.

$$z_{ik}^{t+1} = z_{ik}^t + s_{ik}^{t+1} \tag{14}$$

Where:

ω is the inertia weight

c_1 and c_2 are learning factors

$(l_{ik}^t - z_{ik}^t)$ is the distance between current position and the individual best position

$(l_{gk}^t - z_{ik}^t)$ is the distance between the current position and the global best position

Considering that the search space should be bounded, then it translate that the speed and position are also limited within the same space.

2.5. Adaptive Particle Swarm Optimization (APSO)

APSO is an improved version of PSO. Modified XGBoost has several hyperparameters that need to be tuned and PSO algorithm has the ability of optimizing these hyperparameters at ago in any dimension space. The limitation of the PSO is that it deals with the entire population the same way regardless of their distribution. This issue is curbed by use of APSO. APSO cluster population into various classes but of same distributions and thus preventing local convergence of the particles. The search process in APSO is attained when global optimal value is achieved. The concept of APSO is that every center of the cluster has two features; the center is surrounded with low local density points and larger distance from the high local density points.

Adaptive Particle Swarm Optimization (APSO) divide the entire population (group) of particles into subgroups depending on the distribution characteristic. This is achieved by use of local density and Euclidean distance. The subgrouping of particles helps to manage the exploration and exploitation ability of the PSO algorithm to adapt the behavior of particles locally.

Local Density is Calculated in order to determine how concentrated the particles are within a certain neighborhood of a particular particle. It is used to know regions of low and high density in terms of particles and thus indication of areas of possible optima. Particles are then clustered into subgroups using local density and Euclidean distance.

The breadth $b_{i,j}$ between two particles, i nad j is calculated using the formula below;

$$b_{i,j} = \sqrt{\sum_{k=1}^{n}(z_{ik} - z_{jk})^2} \tag{15}$$

Where z_{ik} and z_{jk} are the position components of particle i and j respectively. A number of particles concentrated within a give distance α is known as the local density is denoted as β_i. It can be estimated as;

$$\beta_i = \sum_{i\neq j} e^{-(\frac{b_{i,j}^2}{\alpha})} \tag{16}$$

APSO adapt the behaviour of a subgroup using three parameters of PSO, inertia weight (ω), cognitive coefficient (c_1) and social coefficient (c_2). Particles of high density indicate chances of local optima and thus can be managed by reducing ω as a way of encouraging exploitation. The opposite is to increase ω on the low density subgroups so as to encourage exploration. To achieve this, the following formula is used.

$$\omega(t) = \omega_{max} - \Big(\frac{\omega_{max} - \omega_{min}}{T_{max}}\Big)t$$

Where T_{max} is the maximum number of iterations and t is time. The dynamics of cognitive and social coefficients helps to balance exploitation and exploration. So;

$$c_1(t) = c_1(t-1) + \Delta c_1$$

and

$$c_2(t) = c_2(t-1) + \Delta c_2$$

Where Δc_1 and Δc_2 depend on the strategies adopted.
and

$$\alpha_i = \min_{j:\beta_j>\beta_i} b_{i,j}$$

Each set of particles in a subset is split into local optimal and ordinary particles depending on the outcome of division of the subgroups. The ordinary particles then exert their search ability and hence the updated formula is;

$$s_i^d = \omega(t)s_i^d + c_1(t)rand_1^d\Big(lbest_i^d - z_i^d\Big) + c_2(t)rand_2^d\Big(cgbest_a^d - z_i^d\Big) \tag{17}$$

and

$$z_i^d = z_i^d + s_i^d \tag{18}$$

Where ω is the inertia weight, c_1 and c_2 are learning factors, $rand_1^d$ and $rand_2^d$ are random numbers in [0,1] interval, $lbest_i^d$ is the best location of particles and $cgbest_a^d$ is the current best position of particles in subgroup a. All the subgroups need to exchange information amongst themselves. This is done through integration of information of each subgroup using the formula below;

$$s_i^d = \omega(t)s_i^d + c_1(t)rand_1^d\Big(lbest_i^d - z_i^d\Big) + c_2(t)rand_2^d\Big(\frac{1}{A}\sum_{c=1} agbest_a^d - z_i^d\Big) \tag{19}$$

and

$$z_i^d = z_i^d + s_i^d \tag{20}$$

2.6. APSO-Modified XGBoost

The study is developing a credit score model that depend on APSO algorithm to optimize hyperparameters of a modified XGBoost. The steps involved are data preparation, feature engineering and model training. Data preparation include dropping feature that are not important like client ID and those with high percentage of missing values. Dataset is also standardized. Feature engineering is about selecting features with high impact on the final results of the model in terms of performance. Using feature importance techniques, variable with low scores and those with also same correlations are dropped. The next step is combining the select features with the tuned hyperparameters using APSO to train the model. This is illustrated on figure 1 below. 1

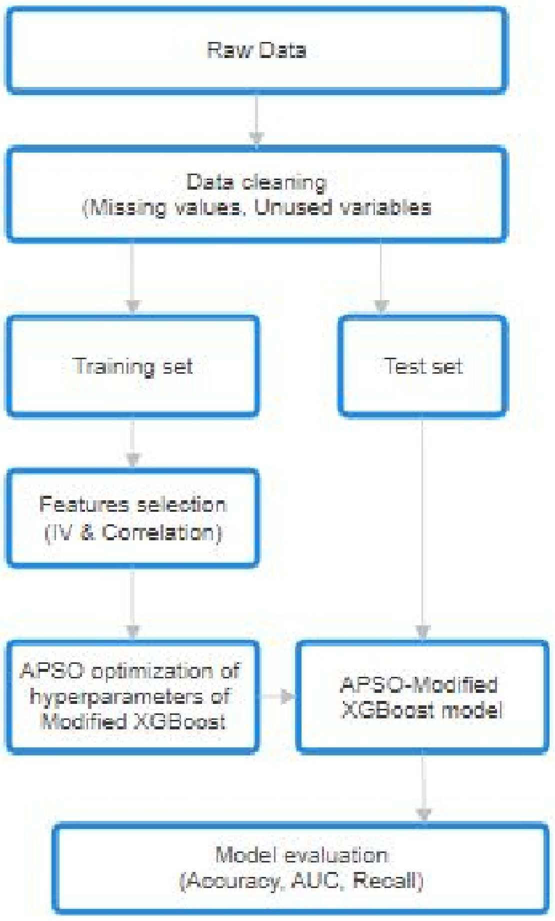
Raw Data
Data cleaning
(Missing values, Unused variables
Training set
Test set
Features selection
(IV & Correlation)
APSO optimization of hyperparameters of Modified XGBoost
APSO-Modified XGBoost model
Model evaluation
(Accuracy, AUC, Recall)

Figure 1. *APSO-Modified XGBoost model process.*

2.6.1. Data scaling

Data scaling is the process of transforming the variable/feature values into a specific range like 0 - 1. There are three commonly used methods of data scaling; standardization is a method that makes the mean of variable equal to zero and its standard deviation equal to one. Normalization method convert the values of each variable to be between the range of zero and one. The last one is min-max method which makes the minimum value of each feature becomes zero and the maximum becomes one. Depending on the problem to be solved and the model to be used, one of the three methods must be used. With the knowledge that APSO algorithm will have to find the euclidean distance at some point and thus the need of using the standardization method to transform the variable values of the data. Let $Z = \{z_1, z_2, ..., z_n\}$ be the input variables and $Y = \{0, 1\}$ be the target variable of the training dataset. Then the values of input variables can be converted into a range of 0 and 1 using the following formula;

$$z_{std} = \frac{z - z_{min}}{z_{max} - z_{min}}$$

Where z_{std} is the standardize value of z.

2.6.2. Important features

In the set of all the features in the dataset, some will not play any role in the model because of their very low predictive ability. Such variables need to be dropped and the process of carrying out that activity is called feature selection. Among the feature selection methods are weight of evidence and information value which are used to measure the predictive power of an independent variable. It can also handle the outliers and missing values. Once the Iv has been calculated, the below table is used to decide whether the variable is to be dropped or not. Using Table 2 below, the features to be used should be falling under the range $0.1 \leq$ IV ¡ 0.3

Table 2. *Information value categories.*

Range	Category
IV < 0.02	Not useful for prediction
$0.02 \leq$ IV < 0.1	Weak predictor
$0.1 \leq$ IV < 0.3	Medium predictor
IV ≥ 0.3	Strong predictor

On the other hand, feature correlation is used to measure the relationship between variables. Two variables that are highly correlated convey similar information and thus leading to multicollinearity. Solution to such cases is dropping one among the two variables. The is achievable by either using correlation matrices or heat-maps for visualization purpose. Figure 2 below shows the correlation of the features used.

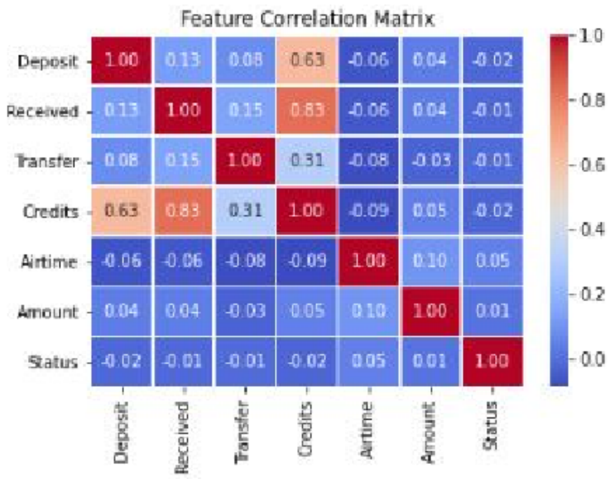

Figure 2. *Feature correlation.*

2.6.3. Model training

Maximum depth, minimum child weight, subsample ratio, column subsample ratio, Inflation rate, learning rate and gamma are among the hyperparameters of modified XGBoost model that need to be optimized in the search space. The particles are clustered into adaptive groups by calculating local density of the particles and distances from particles with higher local density. Considering the particle's position, the hyperparameters of the modified XGBoost is determined and used for prediction. Credit scoring being classification, logistic loss function expressed as below is used.

$$L = \frac{1}{n}\sum_{i=1}^{n}\Big(y_i log(p_i) + (1 - y_i) log(1 - p_i)\Big) \tag{21}$$

The particles are classified as either ordinary or optimum based on their fitness value. The method uses various update strategies to update particle information. If the termination condition is met, the optimal value is obtained. The model reclassifies the population, calculates the fitness value, and updates each particle's position until the termination condition is met. The model is built with optimal hyperparameters and trained and predicted using data. The pseudo-code for the APSO-Modified XGBoost is given below.

Algorithm 1 APSO-based Hyperparameter Optimization with Modified-XGBoost

```
1:  Input: Initialize particles $\mathbf{Z}_j = (z_{j1}, z_{j2}, \ldots, z_{jK})$ with
        position $\mathbf{L}_j = [l_{j1}, l_{j2}, \ldots, l_{jK}]$ and velocity $\mathbf{S}_j = [s_{j1}, s_{j2}, \ldots, s_{jK}]$
2:  Output: Optimal value of hyper-parameter
3:  for $j = 1$ to $N$ do
4:      Compute the local density $\beta_i$ and the distance $\alpha_i$
5:      Choose particles with high $\beta_i$ and relatively high $\alpha_i$ as centers according to $\gamma_i = \alpha_i \cdot \beta_i$
6:      Assign remaining particles and get $A$ subgroups
7:      Initialize XGBoost with instance nodes set $I$ on train data, hyper-parameter $\leftarrow$ current optimal value
8:      for $k = 1$ to $m$ do
9:          gain $\leftarrow 0$, $F = \sum_{i \in I} f_i \leftarrow 0$, $H = \sum_{i \in I} h_i \leftarrow 0$
10:         for $j$ in sorted $(I,$ by $x_{jk})$ do
11:             $F_L \leftarrow F_L + f_L$, $H_L \leftarrow H_L + h_L$
12:             $F_R \leftarrow F - F_L$, $H_R \leftarrow H - H_L$
13:         end for
14:     end for
15:     Update particle state $(pBest_i, gBest)$ referring to the loss function
16:     for $a = 1$ to $A$ do
17:         if particle is local optimal then
18:             (update position and velocity based on APSO equations)
19:         else
20:             $z_j^d = z_j^d + s_j^d$
21:             $s_j^d = \omega s_j^d + c_1 \cdot \text{rand}_1^d \cdot (lBest_j^d - z_j^d) + c_2 \cdot \text{rand}_2^d \cdot \left(\frac{1}{A}\sum_{a=1} cgBest_a^d - z_j^d\right)$
22:         end if
23:     end for
24: end for
```

The description of the algorithm is summarized as follows: First, the initialization process of APSO starts; the particles with position L and velocity S in the search space are initialized as the basic input. Subsequently, the overall adaptive partitioning algorithm starts: the distance α and density β of each particle are calculated, and the particles with high β and α are selected as the centres; the remaining particles are assigned; and the swarm is divided into A subgroups. Modified XGBoost's nodes are initialized using training data, and the model's hyperparameters are based on current optimal values. The modeling process involves calculating each node's gain and splitting the node with the greatest score to create a tree structure. After modeling, the particle type is updated based on the fitness value (loss function), followed by updating two types of particles using different tactics in each subgroup. Following optimization, XGBoost with optimal hyperparameter values was used for testing.

3. Competing Models

The study compares the proposed modified XGBoost model with several models that are discussed in the preceding sub-sections.

3.1. Support Vector Machine (SVM)

The SVM maps an instance's feature vector to a point in space and uses a line to distinguish between two sorts of points. Credit scoring correctly divides data into default and nondefault types.The SVM uses a hyperplane to separate data points. To effectively separate the data, the sum of distances between the closest points on both sides of the hyperplane should be as large as possible.

3.2. Decision Tree (DT)

This is a typical practice in credit scoring.DT classifies instances based on features, with internal nodes representing attribute judgements, branches representing outputs, and leaf nodes representing classification results.The classification outcome in credit score is either default or nondefault.The decision-making method picks the optimal partitioned subtree based on mistake rate and cost of misclassification.

3.3. Logistic Regression (LR)

Logistic regression is a standard statistical technique for binary classification and credit scoring. Regression analysis describes the relationship between independent variables (x) and dependent variables (Y) and predicts the latter. LR adds a logistic function to regression. Credit scoring predicts future results based on prior data from applicants.The purpose of LR is to classify customers' feature vectors as default or nondefault.

3.4. KNN

The K-Nearest Neighbors (KNN) technique is a simple instance-based machine learning model that may be used for classification and regression. It operates by determining the K nearest data points in the feature space to a new input using a distance metric such as Euclidean or Manhattan distance. The model does not go through a training phase; instead, it memorizes the training dataset and predicts using a majority vote (in classification) or averaging of nearest neighbors (in regression). KNN is simple to learn and implement, but it can be computationally expensive and susceptible to noise, particularly in high-dimensional fields. Despite its simplicity, KNN is widely employed as a basis model in pattern recognition, recommendation systems, and anomaly detection.

4. Evaluation Measures

Confusion matrix is one of the best tool of evaluating performance of a model especially when it comes to a classification problem. It has the ability of separating the accurate and inaccurate instances based on the model's predictions.The matrix subgroup the number of instances produced into four subsets. True positive which is model correctly predicted as positive outcome and the actual outcome was positive. True negative is where the model correctly predicted a negative outcome and the actual outcome was negative. False positive (type I error) is where the model incorrectly predicted a positive outcome while the actual outcome was negative. False negative (type II error) is where the model incorrectly predicted a negative outcome but the actual outcome was positive. These four subsets are used calculate accuracy, precision, recall and F1 score of the model. Figure 3 is the confusion matrix for the modified XGBoost.

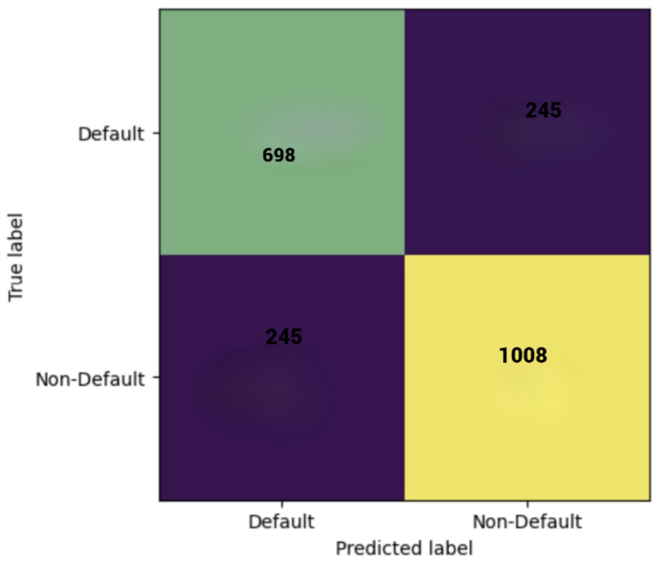

Figure 3. *Confusion Matrix for Modified XGBoost.*

Accuracy is the ability of the model to classify/predict correctly. It is the ratio of total correct instance to the total instances. That is;

$$Accuracy = \frac{TP + FN}{Total}$$

Precision is a measure of how accurate a model's positive prediction are. It is the ratio of True Positive predictions to the total number of positive predictions made by the model. That is from all classes predicted as positive, how many are actually positive.

$$precision = \frac{TP}{TP + FP}$$

Note that precision is used when the interest is minimization of the false positive like in the case of credit analysis where the model should avoid classifying defaulters as non-defaulters.

Recall measure the effectiveness of a model in identifying all relevant samples from a dataset. Mostly used when interested with minimization of false negative like in the cases of medical diagnoses.

$$recall = \frac{TP}{TP + FN}$$

F1-score is the harmonic mean of precision and recall. It display the overall performance of the classification model.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

5. Results and Discussions

The accuracy is one of the most important measures for assessing accuracy of a model. The accuracy gives an indication of the overall predictive ability of the models. In the area of credit scoring a very small improvement in the performance helps financial institution reduce their losses by helping them with a large number of applications at risk of defaulting. Table 2 gives a summary of the models and the modified XGBoost model that was optimized based on the APSO. The result indicates that the modified XGBoost model achieved a promising performance with an accuracy of 89.45%. This was followed by the standard XGBoost with an accuracy of 89.21%. Decision tree was the least perfoming with an accuracy of 83.59%.

Table 3. *Test set comparison metrics.*

Model	Accuracy %	AUC	Precision	Recall	F1 Score
APSO Modified XGBoost	89.45	0.6402	0.5600	0.1850	0.2781
Standard XGBoost	89.21	0.6230	0.5526	0.0925	0.1585
Logistic Regression	89.01	0.5087	0.0000	0.0000	0.0000
Decision Tree	83.59	0.6198	0.2812	0.3172	0.2981
SVM	89.01	0.5048	0.0000	0.0000	0.0000
KNN	88.14	0.5870	0.3125	0.0661	0.1091

Figure 4 gives the ROC curve for the competing models. The area under the curve for the APSO modified XGBoost model was the highest compared to the competing models indicating that the model performs better compared to the competing models.

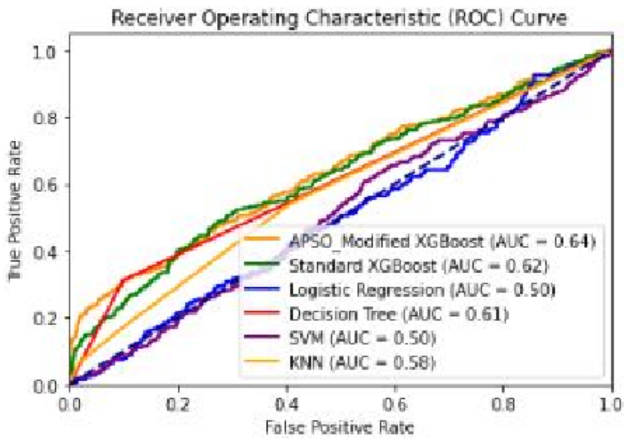

Figure 4. *RoC curves.*

6. Conclusion

The accuracy of a financial credit prediction has a direct effect on the revenue of the financial institution. The purpose of ensemble learning is to be able to gather a series of learners and combine them to form a stronger learner. In this study a modified XGBoost Model is developed that is based on gradient boosting and decision trees. The modified model intergrates the inflation parameter tothe model. In order to be able to obtain the optimal hyperparameters and to improve the accuracy of the model, in this study the data was pre-processed and standardized. After that, feature engineering was used in order to remove features that are redundant. The APSO algorithm was used to optimize the hyperparameters of the model. The results showed that the performance of APSO-Modified XGBoost model improves the credit dataset scoring compared to the performance of the other models the standard XGBoost, logistic regression, decision tree, support vector machine and KNN. Recommendations for future research work is to be able to focus on more features and comparison of models that are different under several optimization techniques.

Acknowledgments

The authors extend their appreciation to their universities for supporting their research work.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. Journal of machine learning research, 13(2), 2012.

[2] Yung-Chia Chang, Kuei-Hu Chang, and Guan-Jhih Wu. Application of extreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. Applied Soft Computing, 73: 914-920, 2018.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785-794, 2016.

[4] Robert A Eisenbeis. Problems in applying discriminant analysis in credit scoring models. Journal of Banking & Finance, 2(3): 205-219, 1978.

[5] Diego Paganoti Fonseca, Peter Fernandes Wanke, and Henrique Luiz Correa. A two-stage fuzzy neural approach for credit risk assessment in a brazilian credit card company. Applied Soft Computing, 92: 106329, 2020.

[6] Akhil Bandhu Hens and Manoj Kumar Tiwari. Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method. Expert Systems with Applications, 39(8): 6774-6781, 2012.

[7] Nan-Chen Hsieh and Lun-Ping Hung. A data driven ensemble classifier for credit scoring analysis. Expert systems with Applications, 37(1): 534-545, 2010.

[8] Hui-Ling Huang and Fang-Lin Chang. Esvm: Evolutionary support vector machine for automatic feature selection and classification of microarray data. Biosystems, 90(2): 516-528, 2007.

[9] Chao Qin, Yunfeng Zhang, Fangxun Bao, Caiming Zhang, Peide Liu, and Peipei Liu. Xgboost optimized by adaptive particle swarm optimization for credit scoring. Mathematical Problems in Engineering, 2021(1): 6655510, 2021.

[10] Kennedy J Eberhart RC et al. Particle swarm optimization. In Proc IEEE Int Conf Neural Networks, volume 4, pages 1942-1948, 1995.

[11] Feng Shen, Xingchao Zhao, Zhiyong Li, Ke Li, and Zhiyi Meng. A novel ensemble classification model based on neural networks and a classifier optimisation technique for imbalanced credit risk evaluation. Physica A: Statistical Mechanics and its Applications, 526: 121073, 2019.

[12] Gang Wang, Jinxing Hao, Jian Ma, and Hongbing Jiang. A comparative assessment of ensemble learning for credit scoring. Expert systems with applications, 38(1): 223-230, 2011.

[13] Gang Wang, Jian Ma, Lihua Huang, and Kaiquan Xu. Two credit scoring models based on dual strategy ensemble trees. Knowledge-Based Systems, 26: 61-68, 2012.

[14] David H Wolpert and William G Macready. No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1): 67-82, 1997.

[15] Yufei Xia, Chuanzhe Liu, and Nana Liu. Cost-sensitive boosted tree for loan evaluation in peer- to-peer lending. Electronic Commerce Research and Applications, 24: 30-49, 2017.

[16] Qiang Yang, Wei-Neng Chen, Jeremiah Da Deng, Yun Li, Tianlong Gu, and Jun Zhang. A level-based learning swarm optimizer for large-scale optimization. IEEE Transactions on Evolutionary Computation, 22(4): 578-594, 2017.

[17] Maciej Zięba, Sebastian K Tomczak, and Jakub M Tomczak. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. Expert systems with applications, 58: 93-101, 2016.